LETTERS

Advantages of Julian Dates

We should use Julian dates instead of the Gregorian calendar in our data bases and programs. I'll explain why, but first I'll clear up a possible confusion regarding the term "Julian date." Many people in our business believe that a Julian date is composed of the year and the day of the year; for them the "Julian date" corresponding to Jan. 1, 1983 (Gregorian) is [19]83.001.

That is not a Julian date. A Julian date has no year in it; it is a serial number associated with a given Gregorian date, serial number 1 having been assigned to a day thousands of years in the past. The Julian date for Jan. 1, 1983 is in fact 2 445 336.

Now for some advantages of using Julian dates. I emphasize that I do not advocate entry or display of Julian dates. I merely urge that all dates be carried internally in data bases and so on in Julian form and converted from and to Gregorian form as necessary.

Many applications require the number of days between two dates or the date some number of days before or after a given date. Most shops have more or less elaborate subroutines to do the calculations. Using Julian dates, the process is simplicity itself — just add or subtract.

It's often necessary to determine the day of the week on which a given date falls. The information is much more easily obtained from the Julian form than from the Gregorian form. Just take Julian date modulo 7. If the result is zero, the day of the week is Monday; if 1, it's Tuesday.

For binary hardware, the Julian date will fit nicely into 32 bits. Indeed, 24 bits will do through Monday, May 9, 41222.

The 21st century looms. Some problems may arise from the widespread practice of carrying only the last two digits of the year. These problems wouldn't arise through the use of Julian dates.

Interconversion of Gregorian and Julian dates is easy. Henry F. Fliegel and Thomas C. Van Flandern published a pair of excellent algorithms in 11 Comm ACM 657 (October 1968). As an exercise in hand optimization, I coded the algorithms as reentrant subroutines for IBM 360/370 CPUs, invocable by Cobol, Fortran, PL/I or assembler callers.

Together, the subroutines occupy about 320 bytes. On an IBM 3033 CPU, Gregorian to Julian conversion takes about 12 microseconds; going the other way takes about 14.

The Fliegel and Van Flandern algorithms are nicely tolerant. For example, if a bank lends on a 180-day note on Nov. 22, 1982, it can find the due date by feeding Nov. 202, 1982 to the Gregorian-Julian algorithm, which will absorb it without a hiccup.

Data base designers might do well to carry Julian dates in the physical data base, accepting and presenting Gregorian dates in the logical counterpart, thus rendering interconversions invisible to the user.

Richard L. Conner
San Francisco, Calif.

(I forgot to attach this to my WP.)

FINAL POSTSCRIPT:
AN ATTRACTIVE (?) ALTERNATIVE TO THE USE OF KEYWORDS IN DATE FUNCTIONS TO INDICATE FORMATS WHERE LENGTH IS THE SAME (YR-MO-DA, MO-DA-YR, DA-MO-YR, etc.) WOULD BE TO ADD A CLAUSE (OR MODIFY PICTURE) TO PROVIDE TO FORMAT INFORMATION WITH THE ITEM ITSELF, E.G. IN PICTURE, THE FOLLOWING CONVENTIONS COULD BE USED:
1 = DAY OF YEAR (1-366)
2 = DAY OF MONTH (1-31)
3 = DAY OF WEEK (1-7)
4 = MONTH (1-12)
5 = YR (00-99)
6 = YEAR (1601-9999)
7
8
THESE SYMBOLS WOULD BE TREATED AS THOUGH THEY WERE 9's OUTSIDE OF DATE FUNCTIONS (& POSSIBLY VALIDATE). DATE DATA TYPES ARE COMMON IN NONPROCEDURAL & MICRO USER TOOLS, SO THIS